Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Software support for signal analysis in new generation of nanopore sequencers

Master's thesis

2024
Ing. Adrián Király

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

Software support for signal analysis in new generation of nanopore sequencers

Master's thesis

Study Programme:	Applied Computer Science
Field of Study:	Computer Science
Department:	Department of Applied Informatics
Supervisor:	doc. Mgr. Tomáš Vinař, PhD.

Bratislava, 2024
Ing. Adrián Király

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Ing. Adrián Király
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Software support for signal analysis in new generation of nanopore sequencers
Softvérová podpora signálovej analýzy pre novú generáciu nanopórového sekvenovania

Anotácia: Cieľom práce je vytvorenie softvérovej podpory pre analýza signálu v novej generácii nanopórových sekvenačných prístrojov a adaptovať existujúce nástroje na novú generáciu nástrojov.

Vedúci: doc. Mgr. Tomáš Vinař, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 01.12.2023

Dátum schválenia: 14.12.2023 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Comenius University Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Ing. Adrián Király
Study programme:	Applied Computer Science (Single degree study, master II. deg., full time form)
Field of Study:	Computer Science
Type of Thesis:	Diploma Thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Software support for signal analysis in new generation of nanopore sequencers

Annotation: The goal of the thesis is to develop a software support for the signal analysis in new generation of nanopore sequencing devices and adapt existing tools to this new generation.

Supervisor:	doc. Mgr. Tomáš Vinař, PhD.
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	doc. RNDr. Tatiana Jajcayová, PhD.
Assigned:	01.12.2023

Approved: 14.12.2023 prof. RNDr. Roman Ďurikovič, PhD.
Guarantor of Study Programme

.....
Student

.....
Supervisor

Acknowledgments: Tu môžete poďakovať školiteľovi, prípadne ďalším osobám, ktoré vám s prácou nejako pomohli, poradili, poskytli dáta a podobne.

Abstrakt

Slovenský abstrakt v rozsahu 100-500 slov, jeden odstavec. Abstrakt stručne sumarizuje výsledky práce. Mal by byť pochopiteľný pre bežného informatika. Nemal by teda využívať skratky, termíny alebo označenie zavedené v práci, okrem tých, ktoré sú všeobecne známe.

Kľúčové slová: jedno, druhé, tretie (prípadne štvrté, piate)

Abstract

Abstract in the English language (translation of the abstract in the Slovak language).

Keywords:

Contents

List of Figures

List of Tables

Introduction

See file `uvod.tex` for information about recommended contents of the Introduction chapter.

Chapter 1

Current State of Nanopore Signal Analysis

Chapter 2

Methodology and Research Methods

The aim of this chapter is to explore in detail a particular DNA nanopore signal processing pipeline developed at the Computational Biology Group at Comenius University Bratislava. The purpose of this pipeline is to produce an alignment of the raw signal data to a known reference genome. Such alignment can be then used for various other applications [2].

As seen in fig. 2.1, the processing is done in several steps, which will be discussed further in this chapter. First, raw signals from the nanopore sequencer have to be acquired and stored. The raw signals are then converted into a DNA sequence, which is then aligned with a reference genome. However, this obtains only an approximate alignment, which is then further improved by a custom alignment algorithm.

The pipeline itself was developed for use with the previous generation of nanopore sequencers. As the new generation of nanopore sequencing technology became gradually available, many of the software tools developed for this generation became obsolete and were replaced by new ones.

Unfortunately, many bioinformatics pipelines (including this one) depended heavily on these tools. This makes it increasingly difficult (or impossible) to use them with new data obtained from next-generation sequencers.

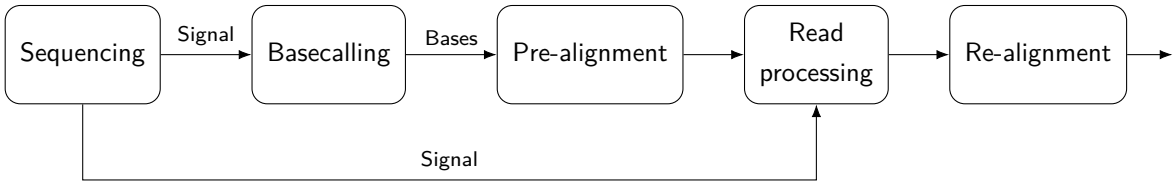

Figure 2.1: A diagram of the nanopore signal alignment pipeline.

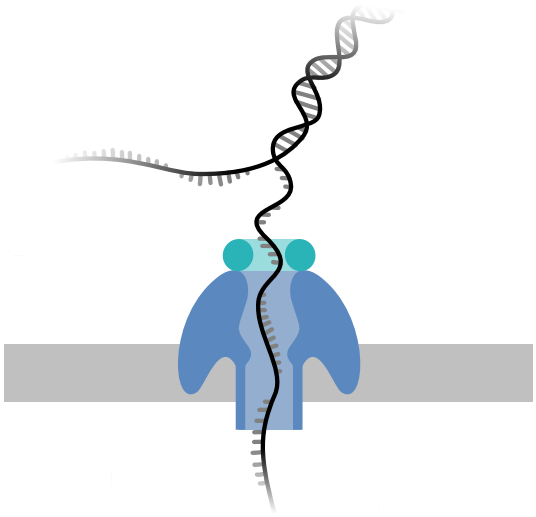

Figure 2.2: Basic principle of nanopore sequencing used by Oxford Nanopore Technologies. A DNA strand is moving through the nanopore, which is used to measure electrical current passing through it.

2.1 Principles of Nanopore Sequencing

The first step of the pipeline is to obtain the data that will be processed. In this case, that means using a DNA sequencer—a device that can determine order of the nucleotide bases of a given DNA. A DNA strand consists of nucleotides, which are composed of one of four different bases: *adenine*, *cytosine*, *guanine*, or *thymine*.

Currently, there are several next-generation sequencing methods in active use and development. One of the most notable methods is the nanopore sequencing developed by Oxford Nanopore Technologies, mostly thanks to its ability to produce very long reads of the strands, its speed, portability, and low cost.

The DNA strand passes through a small barrel-shaped protein, called a nanopore, embedded in a membrane (fig. 2.2). During this process, a special motor protein is attached to the DNA, which regulates the speed of its translocation. By applying voltage across the membrane, it is possible to measure the ionic current passing through the nanopore, which changes as DNA passes through the nanopore [29].

A sequence of raw signal measurements (also known as *squiggle*) can be then used to determine nucleotide bases through a process called *basecalling* (also discussed in section 2.3). Another common practice is to analyze the raw signal data directly (or in combination with basecalling) in an attempt to utilize the additional information present in the signal, or to skip computationally intensive basecalling [2].

A big advantage of nanopore sequencing technology is that the same principles can be used for direct RNA sequencing, whereas other sequencing technologies have to rely on complementary DNA (cDNA) that is synthesized through reverse transcription of

the RNA. Such approach has several disadvantages, for example inability to sequence non-coding RNA [31].

2.2 File Formats for Storage and Processing of the Nanopore Data

During (not just) nanopore sequencing, a large amount of various data is acquired or produced. As such, there are many different data formats designed for different kinds of data, and any data processing pipeline will inherently need to utilize multiple such formats.

This section will describe some of the most used file formats that are relevant to our pipeline—FAST5 for storage of raw nanopore signals, FASTA for representing any nucleotide sequence, and finally, SAM for storing alignments of sequences.

2.2.1 Raw Nanopore Signal

The measurements produced by a nanopore sequencer are transmitted to a computer for further processing, either in real time or asynchronously. However, even if the processing is performed in real time, it is often useful to store the raw signals itself. This makes it possible to perform a different kind of analysis or to repeat the analysis later, with more advanced tools. Furthermore, storing the raw data is necessary for development of new processing techniques.

The amount of data produced can be quite large, as even a single MinION flow cell contains 512 nanopore channels, each of which can simultaneously read a different strand at speeds up to 450 bases per second [13]. To store such amounts of data and to facilitate fast and efficient access, an appropriate data storage format has to be used.

Until recently, all Oxford Nanopore Technologies sequencers used the custom FAST5 format to store the raw signals of reads [6]. The FAST5 format is based on the Hierarchical Data Format (HDF5) designed for efficient storage of large datasets in a structured way.

The HDF5 file logical data model organizes the data using *groups* and *datasets*. A *dataset* is a logical representation of the data itself, with attached metadata such as the data type, size and other properties. Each *group* can store both datasets and other groups, effectively organizing the data into a tree-like structure [27].

The FAST5 format is essentially just a definition of the structure of data contained in a HDF5 file. Originally, a single FAST5 file contained just a single read of the nanopore, but the specification was later changed to allow multiple reads in a single file. This is currently the only supported option, with the single-read files being officially

deprecated and unsupported.

Similarly, the FAST5 format was originally designed to contain not only the raw data from nanopore sequencer, but also results of other analyses performed on this data, such as basecalling. The use of this option is now discouraged, and the ONT tools have removed the support to write the results to FAST5 [15].

In case of the pipeline described in this chapter, the FAST5 files produced by the sequencer are expected to contain only the raw nanopore signal data of multiple reads. These raw signals are stored in FAST5 as a series of 16 bit integers representing quantized values from the analog-to-digital converter of the sequencer. These values can be converted to actual current using the following relation [6]:

$$I_j = (s_i + o)\frac{r}{d}, \tag{2.1}$$

where I_j is the electrical current of the j-th sample in pA, s_j is the quantized value of the j-th sample, o is the offset error of the analog-to-digital converter, r is the measurement range in pA, and d is the number of quantization levels used. The values of o, r, d are stored together with the raw signal data in the FAST5 file.

2.2.2 Nucleotide Base Sequences

FASTA is a simple plain text-based format for the storage of nucleotide sequences. It consists of two parts: a single line starting with the `>` character containing the description of the sequence and the sequence itself stored as a string of letters `A`, `C`, `G`, and `T` (or `U` in RNA) on multiple lines (typically aligned to 80 characters) [11].

Listing 2.1: An example of a sequence stored in the FASTA format.

```
1 >NC_004354.4 Drosophila melanogaster chromosome X
2 GAATTCGTCAGAAATGAGCTAAACAAATTTAAATCATTAAATGCGAGCGGCGAATCCGGAAACAGCAACTTCAAACCAGT
3 CACTCTGGCTGAACTAAATGGCCTGATAAACTCACTGGAATTAAAGAAAGCCCCAGGAACTGACAATCTTAACAACAAGA
4 CCATAATAAACTTACCTACAAAGGCCAGAATATATTTAATACTTATTTATAACAACATCCTGAGAACTGGACATTTCCCG
5 ...
```

Over the years, FASTA became the de facto standard for handling nucleotide sequences, whether as a storage format for reference genomes, input format for sequence alignment, or querying genome databases. Its popularity, simplicity, and lack of any formal standardization have naturally led to the creation of many modifications and variants [4].

One such variant is the FASTQ format, which extends the basic FASTA format with a sequencing quality score of each base. This is achieved by including a *quality line* for each line of the sequence, which contains the same number of characters. These

characters represent different estimated probabilities of error, which serves as a quality score for each base [4].

2.2.3 Sequence Alignment

One of the most common tasks in bioinformatics is the alignment of sequences (discussed further in section 2.4. The primary goal is to rearrange one or more sequences so that shared subsequences are aligned. One of the most used formats for storing results of alignment is the Sequence Alignment Map (SAM) format and its compressed binary equivalent BAM.

SAM is a plain text-based format that consists of header lines containing various (optional) metadata, and multiple alignment lines, each describing one linear alignment of a segment to reference. Each line consists of multiple tab-separated fields, the most important being the position of the leftmost mapping and a *Compact Idiosyncratic Gapped Alignment Report* (CIGAR) string representing the alignment [28].

Op	Description
M	alignment match
I	insertion to the reference
D	deletion from the reference
N	skipped region from the reference

Table 2.1: Overview of basic operations in CIGAR strings.

The CIGAR string is a sequence of operations represented by letters (see table 2.1), prefixed by a number indicating how many times the operation is repeated. For example, `6M2D3M` represents a segment with six bases exactly matched to the reference, two deletions from the reference, and another three exact matches [28].

As mentioned earlier, the binary counterpart to SAM is the BAM format, which has the same internal structure as SAM, but with added gzip compression. To enable fast random access, a BAI file with an index can be created [28].

2.3 Basecalling of the Nanopore Signal Data

As explained in section 2.1, the direct output of nanopore sequencing is a sequence of electrical currents measured across the nanopore, affected by the DNA or RNA strand passing through it. In order to obtain a sequence of nucleotide bases, the raw signal has to be processed first. This process is commonly referred to as *basecalling*.

While sequenced bases are not the desired output of this pipeline, basecalling is used internally for alignment and mapping performed by the Oxford Nanopore Technologies

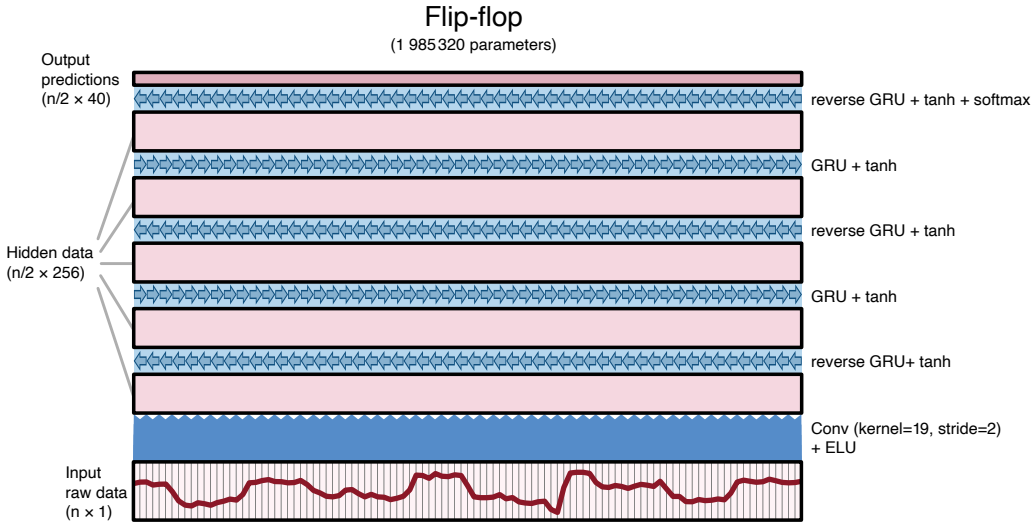

Figure 2.3: The flip-flop neural network architecture used in the Guppy basecaller [30].

tool *Megalodon* as the next step of the pipeline, described in section 2.4 [16].

The nature of the raw signal data makes basecalling (or any other kind of processing) a rather complex task. In addition to the usual issues such as noise in the signal, there are numerous other ones caused by the sequencing process itself.

Notably, a single measurement does not correspond to just one nucleotide, but to a whole sequence of nucleotides (also called *k-mer*) that are currently in the nanopore and affect the measured current [22]. The size of the k-mer depends on the used nanopore chemistry and has slowly increased over the years, with 9-mer models currently being used for basecalling [19].

Furthermore, the motor protein attached to the DNA cannot maintain a completely constant translocation speed, while the sampling frequency remains the same. As a result, signal segments of the same duration do not correspond to the same number of nucleotide bases [22].

Another source of complexity are chemical modifications of nucleotides, such as methylations, where a regular nucleotide is modified by attaching a methyl group. This does not change the resulting DNA sequence, but the different chemical properties of these nucleotides can affect the resulting signal [22]. On the one hand, this makes basecalling much more difficult, but the additional information in the raw signal makes it possible to develop novel methods that can detect these modifications [2, 8].

Even though early basecallers used hidden Markov models [5], the complexity of the task quickly led to the adoption of more powerful approaches such as deep convolutional and recurrent networks [3, 25]. Currently, the research focuses on transformer-based basecallers, which achieved even higher accuracy [18].

Although the architecture of models used in the Guppy basecaller was always based on convolution and recurrent neural networks, there have been multiple changes in the

architecture over time. One of the last architectures used is the *flip-flop* architecture shown in fig. 2.3, which is an evolution of architectures used previously.

In the flip-flop architecture, the length of the input signal is first reduced by the convolutional layer. Then, a series of Gated Recurrent Unit (GRU) hidden layers is used to process the data. Finally, the probabilities of each base are obtained in the softmax layer, which are then used to create the final resulting sequence of nucleotides.

Despite the generally good accuracy of current basecallers, the performance is not always consistent across different types of experiments and genomes. The neural network models do not always generalize sufficiently and there are many cases when a model trained on a dataset from certain species performs significantly worse on a different species [30].

Another important aspect of the neural network basecallers is their speed and computational complexity. Because nanopore sequencing technology allows for real-time sequencing, a significant effort is put into making the basecaller capable of real-time processing [30]. Due to the complexity of the models, this often requires high-performance hardware.

2.4 Approximate Alignment of the Raw Nanopore Signal

In the next part of the pipeline, the Oxford Nanopore tool *Megalodon* is used to obtain an approximate alignment of the raw nanopore signal to the bases of the reference genome. As the data obtained from nanopore sequencers can have a quite high error rate, the alignment requires the use of more sophisticated methods that take these errors into account.

The *Megalodon* first performs a regular basecalling using the Oxford Nanopore basecaller *Guppy*, which translates the raw signal data into a sequence of bases. This sequence is then aligned using the *minimap2* aligner [15]. As the output of the *Guppy* neural network is anchored to the input data, it is then possible to create a direct mapping from the raw signal to reference genome [16].

Most aligners, including *minimap2*, are based on an approach called *seed-and-extend*, first used in BLAST [1]. In this approach, two sequences of nucleotide bases are first divided into (usually overlapping) k-mers, which are then compared across the sequences, obtaining positions of k-mers that are present in both sequences, called *seeds*. This comparison can be performed efficiently by saving all the k-mers and their positions from the first sequence into a hash map, which is then used to look up k-mers from the second sequence [7].

As can be seen in fig. 2.4, the seeds (which represent subsequences where the se-

Figure 2.4: Seed-and-extend alignment of sequences.

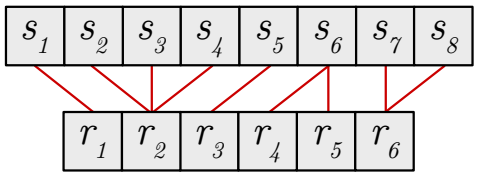

Figure 2.5: Alignment of two sequences using dynamic time warping.

quences match exactly) are joined to other seeds by expanding the subsequences, thus allowing errors in the joined subsequences. Lastly, subsequences that cannot be expanded are joined by adding spaces to the sequences using dynamic programming [7].

The number of possible k-mers grows exponentially with k, so the amount of memory required for the storage of the hash table can be very large for very long sequences and high values of k. A technique called *minimizers* is employed by *minimap2* (and various other aligners) to reduce the number of stored k-mers [7]. This is done by choosing and saving a representative k-mer (usually lexicographically the smallest, or the smallest after applying a hashing function) out of a sliding window of w consecutive k-mers. The same approach is then used for processing of the second sequence [23].

The *minimap2* algorithm also includes numerous improvements and optimizations of this basic process, such as heuristics for expansion and alignment [7]. However, these are out of the scope of this work.

Although this method of alignment is relatively straightforward and has mostly acceptable accuracy, the same kind of complexities that make basecalling a rather difficult task also complicate the alignment. For example, modified bases which often decrese the accuracy of basecalling are one such source of errors [2]. Therefore, if a higher quality of alignment is required, it is necessary to use the additional information contained in the raw signal data to improve accuracy.

2.5 Realigning the Signal Mappings

The last part of the described signal processing pipeline attempts to further improve the signal mappings obtained in the previous section. The method used in this part of the pipeline is inspired by the approach used in the tool *Nadavca* developed by the Computational Biology Group at Comenius University Bratislava [2].

The method itself is based on *dynamic time warping* (DTW), a dynamic programming algorithm that attempts to align two time series in a way that minimizes the difference between their corresponding points by non-linearly shrinking or expanding (warping) the sequences, as can be seen in fig. 2.5 [12].

The alignment of the sequences $s = (s_1, \ldots, s_n)$ and $r = (r_1, \ldots, r_m)$ can be described by an alignment path P:

$$P = (p_1, \ldots, p_l), \tag{2.2}$$

where each tuple $p_i = (n_i, m_i) \in (1, n) \times (1, m)$ represents the alignment of element s_{n_i} to element r_{m_i}. The alignment path P must also satisfy the following conditions[12]:

- boundary condition: $p_1 = (1, 1)$ and $p_l = (n, m)$,
- step-size condition: $p_{i+1} - p_i \in \{(0, 1), (1, 0), (1, 1)\}$ for each successive p_i and p_{i+1}.

The quality of the alignment path P is given as follows:

$$c_P(s, r) = \sum_{i=1}^{|P|} c(s_{n_i}, r_{m_i}), \tag{2.3}$$

where the value of the cost function $c(s_{n_i}, r_{m_i})$ is small if the elements s_{n_i} and r_{m_i} are similar. Then, the optimal alignment path is a path P, for which the value of c_P is the lowest possible. More formally, the cost of the optimal alignment path between sequences r and s is:

$$\mathrm{DTW}(r, s) = \min_{p \in A} \left(c_p \left(r, s \right) \right), \tag{2.4}$$

where A is the set of all alignment paths between r and s that satisfy the conditions mentioned above [12].

To avoid costly and inefficient computation of all possible alignment paths, a dynamic programming approach can be used. In essence, this solution works by constructing an accumulated cost matrix D with dimensions (n, m). Each element $D(i, j)$ has a value that corresponds to the cost of optimal alignment between subsequences $s(1 : i)$ and $r(1 : j)$ [12]:

$$D(i, j) = \mathrm{DTW} \left(s(1 : i), r(1 : j) \right). \tag{2.5}$$

The matrix D can be computed iteratively using the following relation [12]:

$$D(i, j) = \begin{cases} c(1, 1) & i = 1 \wedge j = 1 \\ \infty & i = 1 \vee j = 1 \\ \min(D(i-1, j), D(i, j-1), D(i-1, j-1)) + c(i, j). \end{cases} \tag{2.6}$$

In case of our pipeline, a modified dynamic time warping based algorithm is used to align the raw signal data to the expected signal constructed from the reference genome. First, the raw signal data are normalized with the normalization algorithm from the

Oxford Nanopore Technologies research basecaller *bonito*, which can be described by the following equations [18]:

$$\text{shift} = \max(10, m_{\text{shift}}(Q_a(s) + Q_b(s)) \tag{2.7}$$

$$\text{scale} = \max(1, m_{\text{scale}}(Q_b(s) - Q_a(s)) \tag{2.8}$$

$$s'_i = \frac{s_i - \text{shift}}{\text{scale}}, \tag{2.9}$$

where $Q_a(s)$ and $Q_b(s)$ denote the a-th and b-th quantiles of the raw data sequence s. The default values of the parameters used by *bonito* are $a = 0.2$, $b = 0.9$, $m_{\text{shift}} = 0.51$, and $m_{\text{scale}} = 0.53$ [18].

The normalized raw data pre-aligned to the reference genome are then used to calculate medians for each 5-mer, which are used to construct expected signal of the reference genome (denoted r). The alignment itself is performed by a custom version of dynamic time warping, in which the most notable difference is the modified step-size condition: $p_{i+1} - p_i \in \{(1, 0), (1, 1)\}$ for each successive p_i and p_{i+1}. In other words, this means that the warping is performed only in one direction, by "shrinking" the raw signal s. The accumulated cost matrix D' is then computed using this modified relation:

$$D'(i,j) = \begin{cases} c(1,1) & i = 1 \wedge j = 1 \\ D(i, j-1) + c(i,j) & i = 1 \wedge j \neq 1 \\ \infty & i \neq 1 \wedge j = 1 \\ \min(D(i, j-1), D(i-1, j-1)) + c(i,j). \end{cases} \tag{2.10}$$

As was shown previously, the dynamic programming approach works by constructing an accumulated cost matrix with dimensions (n, m). The length of the nanopore reads can be quite long, which could cause the matrix to be too large to fit into memory. To avoid this, the signal is split into chunks of more manageable sizes, which are processed by DTW independently.

After the accumulated cost matrix is fully calculated, the final alignment is reconstructed iteratively by retracing the optimal choices from the lower right corner of the matrix, obtaining an alignment of the reference genome to the raw data. Finally, all outputs (squiggles, reference, and mapping) for a single read are saved for further use.

Chapter 3

Designing Signal Analysis Software For New Generation of Nanopore Sequencers

The previous chapter described the various parts of a nanopore signal data alignment pipeline, which will be used as a base of this work. Unfortunately, this pipeline has not been maintained for some time, and as such, it is dependent on now obsolete third-party tools, which have to be replaced. The first part of this chapter will focus on changes brought by the new generation of nanopore sequencers and how they affect the pipeline.

In the next part of the chapter, the whole pipeline will be rebuilt using new generation tools. In order to simplify future maintenance, a new framework for building pipelines will be proposed with the aim of abstracting the implementation details of third-party tools from the pipeline.

Lastly, the capabilities of the signal alignment pipeline will be extended for the alignment of RNA sequences obtained using direct RNA sequencing.

3.1 Changes in the New Generation of Nanopore Sequencers

Since the first commercially available nanopore sequencer was released in 2014 by Oxford Nanopore Technologies, the nanopore sequencing platform has been in active development with numerous changes both in the hardware and software [29]. These changes often enable new functionality or improve the performance and accuracy of sequencing. Unfortunately, many of these changes have been large enough to break compatibility of existing user tools, workflows, and pipelines, such as the one described in chapter 2.

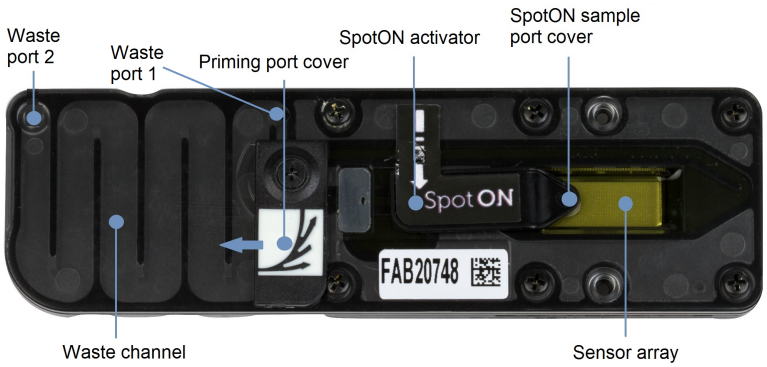

Figure 3.1: A single MinION flow cell consists of the sensor array, which contains the nanopores and circuits for measurment of the currents during sequencing. Samples are added to the chamber of the sensor array directly through a priming port. After sequencing, samples flow to the waste chamber [17].

This has been the case in the past few years with the introduction of the R10 flow cell, new software tools and formats. Although R10 has been released into early access in 2019 [20], the previous generation has been still in use until recently. This section will explain all of the relevant changes that need to be addressed during the development of signal analysis software for this generation of nanopore sequencers.

3.1.1 New Flow Cells and Chemistry Kits

The nanopore is a protein with limited lifespan that decreases mainly through use. As such, the nanopores are not part of the sequencing device itself but are contained in a separate, replacable cartrige called *flow cell* [9]. When the nanopores in the flow cell are spent, the entire flow cell is removed from the sequencer and replaced by a new one right before the next sequencing.

The flow cell (see fig. 3.1) houses an array of nanopores embedded in a polymer membrane in a well of the sensor array. The sensor array is connected to an ASIC that measures currents in the nanopores and controls the sensor array itself [17].

Before DNA or RNA samples can be sequenced, they must first be extracted and chemically prepared by an appropriate procedure (protocol), also known as library preparation. These procedures and chemistry are specific to the type of sequencing performed and the version of the flow cells.

The R10 series of flow cells uses a new design of nanopores with a longer barrel and significantly increases accuracy [10]. Furthermore, the new pores support duplex basecalling, in which the complement strand of the DNA is read immediately after sequencing the template strand. When combined with a proper processing (introduced in the new-generation basecaller Dorado, see section 3.1.3), this further increases the

accuracy of sequencing.

In general, changes in the R10 series have notably improved the accuracy of reads, with some benchmarks reporting an approximately 10 % improvement over the R9 series [13]. The accuracy is now reaching a level where it is starting to be possible to reconstruct the whole genome using only a nanopore sequencer, although hybrid approaches are still considered superior [24].

In some applications, such as detection of methylated bases, R10 flow cells do not seem to provide any observable improvement over the older generation [13]. The R10 series is also supposed to have higher yields, though this was not confirmed as an issue with motor protein has probably caused lower yields than R9 [13, 24].

3.1.2 Data Storage Format

A notable change that is mostly related to the new basecaller (see section 3.1.3) is the introduction of the POD5 data storage format, which replaces the formerly used FAST5 (described in section 2.2.1) with the aim of improving write and read speeds, recovery in case of a crash during writing and increasing the storage efficiency [21].

Similarly to FAST5, POD5 is also not a completely custom format, as it is based on the Apache Arrow project, which defines a language-agnostic in-memory data format designed for fast data interchange and analytics [26]. However, unlike FAST5, it is used only for the raw signal data itself and not for the outputs of basecalling or other analyses [21].

A POD5 file is essentially a container for three Apache Arrow tables [21]:

- a *reads* table which contains metadata for each read performed during the sequencing,
- a *signal* table containing the raw signal data itself,
- a *run info* table identifying the sequencer runs (if the file is a result of multiple runs).

Even though the one of the major motivations behind the POD5 format was to enable performance improvements in the future basecaller (as described in section 3.1.3, the format itself was released before its introduction. As such, support for POD5 was first added to the previous generation basecaller *Guppy* [15].

However, the introduction of the new data format has certainly brought some challenges, as not all legacy software tools have added the support for POD5, making it harder to use newly sequenced data with such tools. While the conversion is a relatively straightforward process, it may cause additional issues mainly in tools that relied on features such as writing basecaller output directly to the FAST5 file.

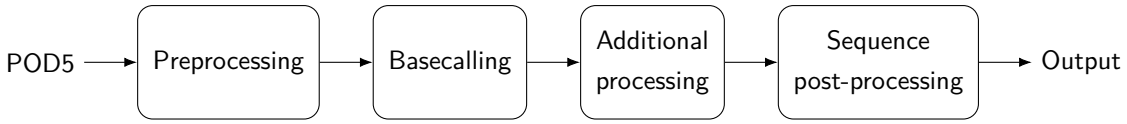

Figure 3.2: The architecture of the Dorado basecalling pipeline.

Fortunately, the impact of the new format on the signal alignment pipeline should be minimal, as no stage of the pipeline expects the FAST5 file to contain anything other than the raw signal. This means that conversion of FAST5 to POD5 files should not cause unexpected problems.

3.1.3 Software Tools

The R10 series of flow cell has been released gradually over the span of several years while in continuous development. Therefore, for a significant amount of time, software support for this generation was developed within previous generation tools. As such, the Guppy basecaller supports some of the older versions of the R10 flow cells [15].

However, in order to fully utilize the capabilities of the updated hardware, a new basecaller (and other tooling) was eventually introduced. A new basecaller *Dorado* was developed from scratch in order to solve many of the issues that the Guppy basecaller had, and to serve as a platform for future developments [14]. It is also noteworthy that unlike ONT's previous basecallers, Dorado and its machine learning models are open-source.

Although *Dorado* is first and foremost a basecaller, it also consists of multiple tools designed for processing nanopore sequencing data, which can be integrated into its basecalling pipeline (described in fig. 3.2).

In the preprocessing part of the basecalling pipeline, the raw signal data is normalized. The choice of normalization method varies with the used basecalling model, with several strategies such as quantile normalization and median absolute deviation commonly used [14]. Additionally, sequencing adapters added to the DNA or RNA strand during library preparation are usually detected and trimmed from the raw signal during this phase [14].

The next step of the pipeline is the basecalling itself. In this part, a machine learning model trained for the specific combination of the nanopore type, the chemistry used, and the type of sequenced strand (DNA or RNA) is used to predict the most probable sequence of bases captured by the signal. The models themselves were initially of an LSTM based architecture similar to the models in Guppy, but recently new transformer-based models were released [14].

The additional processing step of the pipeline provides options for running additional analyzes on the results. For example, the basecalled sequence can be aligned

using an embedded *minimap2* aligner to known reference genome (as described in section 2.4). Finally, the post-processing step can be used for further read trimming and splitting before writing the output [14].

Apart from standard DNA and RNA basecalling, *Dorado* also enables basecalling of modified bases and barcoding—a technique, where special sequences called *barcodes* are attached at the beginning of the strands in order to distinguish samples from multiple sources after sequencing them in same run.

The *Dorado* basecaller is a production basecaller meant for general use on sequenced data. However, ONT also maintains an open source research basecaller *Bonito* used primarily to train new basecaller models and develop new methods. Although Bonito's functionality is limited compared to Dorado, Bonito contains the source code for the models used in Dorado, making it useful for creating new custom models [18].

Changes in the Dorado base caller will have a significant impact on the pipeline. Previously, the re-alignment step of the pipeline relied on the signal mappings generated by Megalodon and the Guppy basecaller, both of which have been deprecated. Therefore, this functionality will have to be replicated using the new Dorado basecaller.

3.2 Designing Signal Analysis Framework

Historically, these types of changes in the nanopore sequencing platform have occurred many times. Although big platform changes such as this one are more rare, maintaining a larger number of pipelines or software tools may require significant effort.

This is in large part due to the way most of these tools are created. Usually, they consist of a number of scripts and are highly dependent on many third-party tools and libraries. This means that even small changes in the capabilities and interfaces of these dependencies cause significant issues in the whole tool.

While reducing the number of dependencies would certainly help in many instances, it is often not possible due to their complexity. However, these issues can be mitigated by using a common framework that abstracts the details of the third-party tools and provides a stable interface for creating own pipelines and tools. The aim of this chapter is to explore the requirements on such framework and design a basic version, which will be used as a basis of the pipeline described in chapter 2.

3.3 Adapting Alignment Pipeline

3.4 Extending Capability for RNA Alignment

Chapter 4

Results

4.1 Designing Evaluation Metrics and Methods

4.2 Performance Evaluation of DNA Re-alignment

4.3 Performance Evaluation of RNA Re-alignment

Conclusion

See file `zaver.tex` for information about recommended contents of the Conclusion chapter.

Bibliography

[1] Altschul, S. F.; Gish, W.; Miller, W.; Myers, E. W. and Lipman, D. J. Basic local alignment search tool, vol. 215, no. 3, p. 403–410. ISSN 00222836. Available at: https://linkinghub.elsevier.com/retrieve/pii/S0022283605803602.

[2] Bož̌a, V.; Batmendijn, E.; Pereš́ini, P.; Hodorová, V.; Lichancová, H. et al. Precise Nanopore Signal Modeling Improves Unsupervised Single-Molecule Methylation Detection, p. 2023–07. Available at: https://www.biorxiv.org/content/10.1101/2023.07.13.548926.abstract.

[3] Bož̌a, V.; Brejová, B. and Vinař, T. DeepNano: Deep recurrent neural networks for base calling in MinION nanopore reads, vol. 12, no. 6, p. e0178751. ISSN 1932-6203. Available at: https://dx.plos.org/10.1371/journal.pone.0178751.

[4] Cock, P. J. A.; Fields, C. J.; Goto, N.; Heuer, M. L. and Rice, P. M. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants, vol. 38, no. 6, p. 1767–1771. ISSN 0305-1048, 1362-4962. Available at: https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkp1137.

[5] David, M.; Dursi, L. J.; Yao, D.; Boutros, P. C. and Simpson, J. T. Nanocall: an open source basecaller for Oxford Nanopore sequencing data, vol. 33, no. 1, p. 49–55. ISSN 1367-4803, 1367-4811. Available at: https://academic.oup.com/bioinformatics/article/33/1/49/2525680.

[6] Gamaarachchi, H.; Samarakoon, H.; Jenner, S. P.; Ferguson, J. M.; Amos, T. G. et al. Fast nanopore sequencing data analysis with SLOW5, vol. 40, no. 7, p. 1026–1029. ISSN 1087-0156, 1546-1696. Available at: https://www.nature.com/articles/s41587-021-01147-4.

[7] Li, H. Minimap2: pairwise alignment for nucleotide sequences, vol. 34, no. 18, p. 3094–3100. ISSN 1367-4803, 1367-4811. Available at: https://academic.oup.com/bioinformatics/article/34/18/3094/4994778.

[8] Liu, Y.; Rosikiewicz, W.; Pan, Z.; Jillette, N.; Wang, P. et al. DNA methylation-calling tools for Oxford Nanopore sequencing: a survey and human epigenome-wide evaluation, vol. 22, no. 1, p. 295. ISSN 1474-760X. Available at: https://doi.org/10.1186/s13059-021-02510-z.

[9] MacKenzie, M. and Argyropoulos, C. An Introduction to Nanopore Sequencing: Past, Present, and Future Considerations, vol. 14, no. 2, p. 459. ISSN 2072-666X. Available at: https://www.mdpi.com/2072-666X/14/2/459.

[10] Mostafa, H. H. An evolution of Nanopore next-generation sequencing technology: implications for medical microbiology and public health, vol. 62, no. 5, p. e00246–24. ISSN 0095-1137, 1098-660X. Available at: https://journals.asm.org/doi/10.1128/jcm.00246-24.

[11] Mount, D. W. *Bioinformatics: sequence and genome analysis*. Cold Spring Harbor Laboratory Press. ISBN 9780879696559 9780879695972 9780879696085.

[12] Müller, M. *Information retrieval for music and motion*. Springer. ISBN 9783540740476. OCLC: ocn172976996.

[13] Ni, Y.; Liu, X.; Simeneh, Z. M.; Yang, M. and Li, R. Benchmarking of Nanopore R10.4 and R9.4.1 flow cells in single-cell whole-genome amplification and whole-genome shotgun sequencing, vol. 21, p. 2352–2364. ISSN 20010370. Available at: https://linkinghub.elsevier.com/retrieve/pii/S2001037023001368.

[14] Oxford Nanopore Technologies. *Dorado 0.9.1 Documentation*. Available at: https://dorado-docs.readthedocs.io/en/0.9.1/.

[15] Oxford Nanopore Technologies. *Guppy protocol*. Available at: https://nanoporetech.com/document/Guppy-protocol.

[16] Oxford Nanopore Technologies. *Megalodon Algorithm Details — Megalodon 2.3.3 documentation*. Available at: https://nanoporetech.github.io/megalodon/algorithm_details.html.

[17] Oxford Nanopore Technologies. *MinION and flow cells*. Available at: https://nanoporetech.com/document/hardware.

[18] Oxford Nanopore Technologies. *Nanoporetech/bonito: A PyTorch Basecaller for Oxford Nanopore Reads*. Available at: https://github.com/nanoporetech/bonito.

[19] Oxford Nanopore Technologies. *Nanoporetech/kmer_models*. Oxford Nanopore Technologies. Available at: https://github.com/nanoporetech/kmer_models. Original-date: 2016-05-06T10:07:01Z.

[20] Oxford Nanopore Technologies. *New 'R10' nanopore released into early access.* Available at: https://nanoporetech.com/news/news-new-r10-nanopore-released-early-access.

[21] Oxford Nanopore Technologies. *Pod5 File Format Documentation.* Available at: https://pod5-file-format.readthedocs.io/en/latest/.

[22] Rang, F. J.; Kloosterman, W. P. and Ridder, J. de. From squiggle to basepair: computational approaches for improving nanopore sequencing read accuracy, vol. 19, no. 1, p. 90. ISSN 1474-760X. Available at: https://doi.org/10.1186/s13059-018-1462-9.

[23] Roberts, M.; Hayes, W.; Hunt, B. R.; Mount, S. M. and Yorke, J. A. Reducing storage requirements for biological sequence comparison, vol. 20, no. 18, p. 3363–3369. ISSN 1367-4811, 1367-4803. Available at: https://academic.oup.com/bioinformatics/article/20/18/3363/202143.

[24] Sanderson, N. D.; Kapel, N.; Rodger, G.; Webster, H.; Lipworth, S. et al. Comparison of R9.4.1/Kit10 and R10/Kit12 Oxford Nanopore flowcells and chemistries in bacterial genome reconstruction, vol. 9, no. 1. ISSN 2057-5858. Available at: https://www.microbiologyresearch.org/content/journal/mgen/10.1099/mgen.0.000910.

[25] Teng, H.; Cao, M. D.; Hall, M. B.; Duarte, T.; Wang, S. et al. Chiron: translating nanopore raw signal directly into nucleotide sequence using deep learning, vol. 7, no. 5, p. giy037. ISSN 2047-217X. Available at: https://academic.oup.com/gigascience/article/doi/10.1093/gigascience/giy037/4966989.

[26] The Apache Software Foundation. *Apache Arrow.* Available at: https://arrow.apache.org/docs/.

[27] The HDF Group. *HDF5 Documentation.* Available at: https://support.hdfgroup.org/releases/hdf5/v1_14/v1_14_5/documentation/doxygen/.

[28] The SAM/BAM Format Specification Working Group. *Sequence Alignment/Map Format Specification.* Available at: http://samtools.github.io/hts-specs/SAMv1.pdf.

[29] Wang, Y.; Zhao, Y.; Bollas, A.; Wang, Y. and Au, K. F. Nanopore sequencing technology, bioinformatics and applications, vol. 39, no. 11, p. 1348–1365. ISSN 1546-1696. Available at: https://www.nature.com/articles/s41587-021-01108-x.

[30] Wick, R. R.; Judd, L. M. and Holt, K. E. Performance of neural network basecalling tools for Oxford Nanopore sequencing, vol. 20, no. 1, p. 129. ISSN 1474-760X. Available at: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-019-1727-y.

[31] Wongsurawat, T.; Jenjaroenpun, P.; Wanchai, V. and Nookaew, I. Native RNA or cDNA Sequencing for Transcriptomic Analysis: A Case Study on Saccharomyces cerevisiae, vol. 10, p. 842299. ISSN 2296-4185. Available at: https://www.frontiersin.org/articles/10.3389/fbioe.2022.842299/full.